

Padawan: An Internet That Ends

Hareesh Nagaraj
hareesh.nagaraj@gmail.com
outside.padawan.nyc

Working draft, September 2026. The system and this paper are beta and under active development; feedback is welcome.

Abstract. A purely personal agent would read the internet on its owner’s behalf and come back with something finished enough to use, without the owner touching a feed. Models with web search provide part of the solution, but the main benefit is lost if the text is still unaccountable: a model with search is a feed with better prose. We describe a system in which no sentence reaches a reader until its claims have survived attack. Discovery lanes gather live sources; a binder turns them into explicit claims; an adversary attacks each claim with a test that could kill it; the tests are run against the live web; each claim ends active, killed, strengthened, or weakened. Every run writes a card recording what ran, what was found, what was attacked, and what survived, and a publication gate refuses any page not drafted from a completed card. The card is the genotype; the published brief is its expression. The system requires minimal structure: one machine, one operator, scheduled jobs that fail loudly and retry. What survives is proof not of eloquence but of having been attacked and left standing.

1. Introduction

Reading on the internet has come to rely almost exclusively on feeds, and a feed’s incentive is retention, not completion. The cost is familiar: unbounded scroll, engagement-ranked claims, and no way to know whether a given sentence would survive five minutes of checking. Large models with web search change the texture but not the structure. They generate fluent summaries whose individual claims carry no history: no record of where a claim came from, who attacked it, or what evidence answered the attack. Trust is asked for rather than earned, and a certain percentage of confident error is accepted as unavoidable.

What is needed is a reading system based on selection pressure instead of trust: one in which text is cheap, and a claim becomes expensive only by surviving. Claims that have been bound to named sources, attacked by a critic that must specify a falsifying test, and retested against the live web can be delivered with their history attached, and the reader can put the phone down when the document ends. In this paper we describe Padawan, a personal system built and operated by one person, and Sapith, the engine inside it that makes research claims earn their place. We define the system’s parts, specify the protocol a claim passes through, describe the receipt each run produces, and state plainly which mechanisms exist, which are deliberately unwired, and which do not exist yet.

2. Definitions

The system uses names from a familiar apprenticeship. The Jedi is the operator: the person the system serves, who sets what it follows and holds the only keys. A Padawan is a Jedi’s apprentice, and the name states the intent: an agent in training that acts on its operator’s behalf and is not yet trusted to open doors on its own. A Holocron is the apprentice’s repository of recorded knowledge.

We define **Padawan** as the whole system: software on one computer that goes out on the operator's behalf, reads the live internet, and returns finished objects. The reading surfaces (a private paper and one public page [6]) are outputs of the system, not the system. Under them sits a factory: lanes, an engine, a memory, and a bar below which nothing ships.

We define **Sapith** as the research engine: the machinery that turns a question into surviving claims. Sapith is not a wrapper around several models. It is a protocol in which roles exchange material under receipts, described in sections 4 through 7.

We define **Hermes** as the hands: the runtime that schedules jobs, executes commands, and delivers messages. Hermes decides nothing about content. The division is fixed: Hermes is the hands, Padawan is the authority on what may ship, Sapith is the evidence engine [5].

We define the **Holocron** as the system's long-term memory store. Each research run projects a ledger of its claims and their fates into it, and later runs recall those ledgers before they begin (section 8).

Four smaller terms recur. A **lane** is one subject the system keeps reading on a schedule. A **charter** is one job's contract: the question, the stakes, the budget, and the stopping condition. A **card** is the receipt a run writes (section 7). A **brief** is a finished, ending document delivered to a reader; the operator's private briefs and the public neighborhood pages are both briefs.

3. Claims

We define a claim as a statement bound to named sources, carrying support edges, preserved conflicts, and a fate. Binding is the step that makes text attackable: a paragraph cannot be killed, but a claim with a subject, a value, and a source can be. Claim identity is a normalized content hash, so the same assertion found twice through different wording is the same claim and inherits its history. Conflicts between sources are preserved on the claim rather than resolved by prose, because a smoothed conflict is information destroyed before the adversary could use it.

4. The Protocol

Every run executes one state machine with eight stages. A charter enters at P0, where boundary checks run and a scheduler fixes the run's knobs. R1 performs live discovery on two concurrent lanes, one over the web and source-of-record pages, one over X, with a wide sweep of at least twelve searches on the social lane. R1b is the crossing: material from each lane is paired against the other, agreements and contradictions are annotated as evidence neither lane alone could produce, and at most one combination-only lead may be chased under a gate. S1 binds sources into claims. A1 attacks them. R2 runs the adversary's tests live. S2 applies the results as fates. SC scores the run, screens its structure, writes the card, and stamps the contract verdict.

```

charter -> P0 -> R1 ---- web lane --+
                        |               +- R1b crossing -> S1 bind
                        +--- X lane  ----+

```

```

S1 claims -> A1 attack -> R2 live retest -> S2 fates -> SC card

```

The eight stages. Every arrow leaves a receipt in the run store.

Two properties of the pipeline matter more than its shape. First, every stage leaves a receipt, so the card can later prove not only what was concluded but what each role actually did. Second, degradation is visible: a stage skipped for budget is recorded as skipped, and a run that cannot complete its stages fails its contract rather than shipping a thinner result under the same name.

5. The Adversary

The adversary is what separates this system from an ensemble. A1 receives the bound claims and produces critiques, and a critique without a falsifying test is not accepted; the protocol raises rather than continuing. A falsifying test is concrete: name the interval, open the calendar page, kill the claim if the date falls outside it. The strongest critiques become tickets, capped at two per run so that attack remains focused rather than performative, and R2 answers each ticket against the live web. The test is run, not recalled: the value of the stage is that the answer comes from the page as it exists now, not from what any model remembers about it.

6. Fates

After R2, every claim ends in exactly one of four states: active, killed, strengthened, or weakened. Fates are derived deterministically from the observed transitions between binding and application; what a model says about its own changes is kept as commentary and trusted for nothing. Killed claims stay named on the card. A system that deleted its failures would be a feed with receipts.

One claim from a production run in Chicago [8] shows the full path. The binder inferred that an exhibition opening August 29 fell inside a two-week window beginning August 14, and preserved a conflict with a source dating a second exhibition's closing to August 26. The adversary attacked the arithmetic and named the test: fix the window's endpoints and check the museum's calendar. The ticket ran live; the window closed August 27. The first claim was killed and the second strengthened. Confidence in the killed claim rose while its load-bearing flag flipped false, which is the intended shape: the system became more certain and less willing to recommend. The brief inherited the survivor, and a plausible mistake never reached the reader.

7. The Card

Every run writes one card: a JSON record of the charter, the scheduler's decisions, every source, every claim with its fate, every critique and ticket and live answer, stage timings, provider execution, structural warnings, and residual risk. The card is then judged against a contract of six requirements. Two of them block: the stages must have completed, and there must be something to publish. The other four (claims survived, live radar ran, the full path executed, no placeholder sources) stamp the card visibly when they fail instead of suppressing it, because a stamped failure teaches and a suppressed one repeats.

The card is the genotype and the brief is the phenotype. Any number of surfaces can be expressed from one card for different audiences, but no surface may cite what the card does not contain. Cards persist; briefs are disposable.

8. Memory

Runs remember. Each run projects a ledger into the Holocron: its claims with fates, and each killed or weakened claim paired with the critique that attacked it and the verdict that answered it. A new run recalls up to five prior ledgers for its charter, at most twelve claims each, and receives them as an advisory block with one strict property: recall cannot add a claim; it can only stop a dead one from being reborn as a fact. Alongside the ledgers, tactics that scored well are promoted per charter and lane and loaded before discovery. Recall fails open; a run with no memory available proceeds and says so on its card.

9. The Source Genome

Sources should earn standing the way claims earn their place: a source that carried surviving claims should be promoted, a source found only under killed claims should decay, and future discovery should start from that standing. The mechanism exists as a pure function over cards, and domain identity is enforced at the door: a run must name one of a fixed registry of legal domains or its results are unwritable, so no shared junk bucket forms. The write path, however, is deliberately unwired. Measured cards showed promotions without a single decay, and standing that only ever rises is flattery, not selection. The decision on record is to wire one domain, judge it blind against a frozen control, and only then let the genome learn. Until that trial runs, the system's between-run inheritance is claims and tactics, not source standing, and this paper says so rather than implying otherwise.

10. Operations

The steps to run the system are as follows:

1. A scheduled job fires with a charter: question, lane, domain, stakes, budget.
2. The engine runs as one process; roles are filled by whichever providers the charter names, and the card records what actually ran.
3. The card is written and judged against its contract.
4. A drafting step turns the card, and only the card, into a brief for its audience.
5. The publication gate checks the receipts: no completed card, no page. A failed job stays at the front of its queue and retries.
6. Delivery surfaces (the paper, the public page, the operator's phone) express the result; every document ends.

Three operating rules hold the system honest. Roles, never brands: discovery, binding, audit, and synthesis are roles, and models are interchangeable only if the role's contract is preserved and recorded. Providers run as local subscription processes with least privilege; the audit role runs sandboxed and read-only with credentials stripped, and mock providers run only when named explicitly, so a quiet fallback to fake research is structurally impossible. Failure must be loud: this rule was earned in production when a runtime upgrade silently broke background processes and one day's pages shipped from a single discovery pass without the engine. The gate that resulted, no card, no page, chose visible staleness over invisible thinness, and its first enforced run published two neighborhoods and correctly refused a third.

11. Limits

We consider what the receipts still cannot prove. A card can show that live discovery ran, sources were real, stages completed, and claims survived, and still not prove the run researched the intended object: one production run passed its full contract while much of its discovery followed the wrong sense of an ambiguous title. The designed control is a semantic scope on charter and card, naming the object under review, allowed and excluded senses, and a scope verdict; it does not exist in code yet, and until it does, on-topic-ness rests on charter wording. Discovery width is a second limit: two lanes exist, and a third lane whose job is disagreement has a pre-registered bar to clear before it may be built. Of the claims that are load-bearing after fates are applied, what fraction were materially changed by a disagreement that lane introduced, and at what cost per change? If the answer is zero across real charters, the lane is noise and is not built. Finally, most of what

this paper calls built is proven by receipts and code, not by blind comparison; the engine’s advantage over a plain model with search is argued by construction and by worked examples, and a controlled benchmark remains open work.

12. Conclusion

We have described a personal system for reading the internet without trusting it. We started with the usual framework of an agent with web search, which provides fluency but no accountability, and added selection pressure: claims bound to sources, an adversary obligated to name the test that would kill each claim, live retests, recorded fates, and a receipt for every run. A publication gate makes the receipts binding, memory keeps dead claims dead, and the parts that do not yet exist are stated as absent rather than assumed. The system requires minimal structure: one machine, one operator, jobs that retry, and rules enforced by refusal rather than intention. The reader’s experience of all this machinery is deliberately small. A document arrives, every sentence in it has already survived an attack, and it ends.

Everything here is a work in progress: the system, its rules, and this description of them. Corrections and disagreement are wanted; write to the address above.

References

- [1] “Sapith: end-state design,” internal spec, Aug 2026: the stage graph and claim-survival thesis.
- [2] “Triple Helix,” operator capture, Aug 2026: the original crossing proposal, since shipped.
- [3] “R1b compounding,” internal spec, Aug 2026: bounds for the crossing stage.
- [4] The workflow-card schema: the run receipt and its six-requirement contract.
- [5] The Padawan invariants ledger, 2026: the enforced rules cited through this paper.
- [6] “What Padawan is,” canonical self-description, Aug 2026: the factory under the paper.
- [7] The source-graph spec, 2026: standing graphs per domain; the closed seeds store of section 9.
- [8] Workflow card, Stopscrolling Chicago run, Aug 2026: the killed-claim example of section 6.